

Functional vs. Nonfunctional Requirements

Functional Requirements

- ◆ Describe user tasks that the system needs to support
- ◆ Phrased as actions
 - “Advertise a new league”
 - “Schedule tournament”
 - “Notify an interest group”

Nonfunctional Requirements

- ◆ Describe properties of the system or the domain
- ◆ Phrased as constraints or negative assertions
 - “All user inputs should be acknowledged within 1 second”
 - “A system crash should not result in data loss”.

Type: functional vs. non-functional

» requirement, functional

A statement of some function or feature that should be implemented in a system [Sommerville 2011].

what?

- **requirement, non-functional**

A statement of a constraint <...> that applies to a system
[Sommerville 2011]

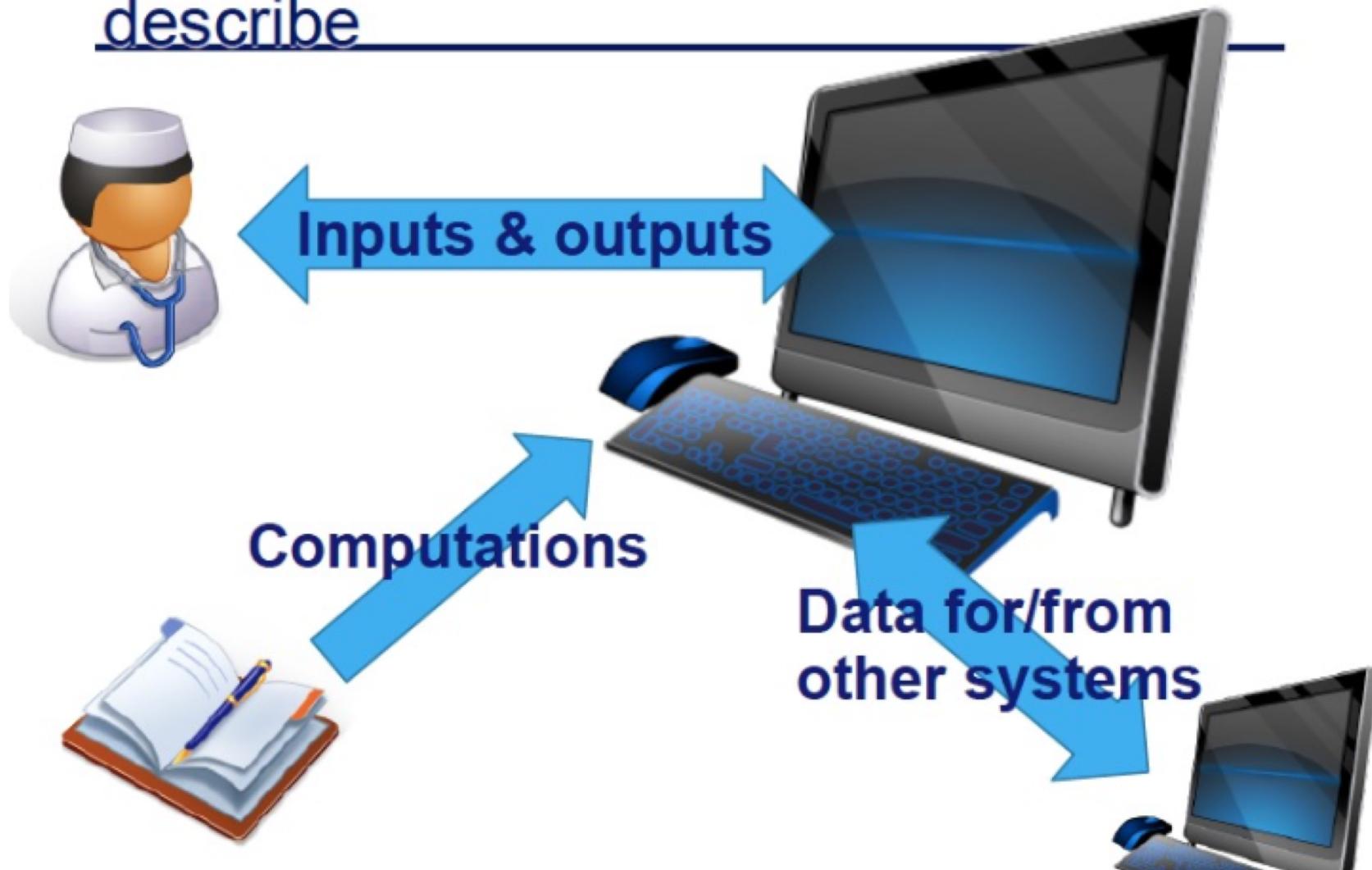
how fast?

how many failures?

how accurate?

how secure?

Functional requirements frequently describe



Non-functional requirements

- » Non-functional requirement relates to quality attributes: e.g., **performance, learnability, availability**
- » functional requirement: "*when the user presses the green button the Options dialog appears*":
 - performance: how quickly the dialog appears;
 - availability: how often this function may fail, and how quickly it should be repaired;
 - learnability: how easy it is to learn this function.

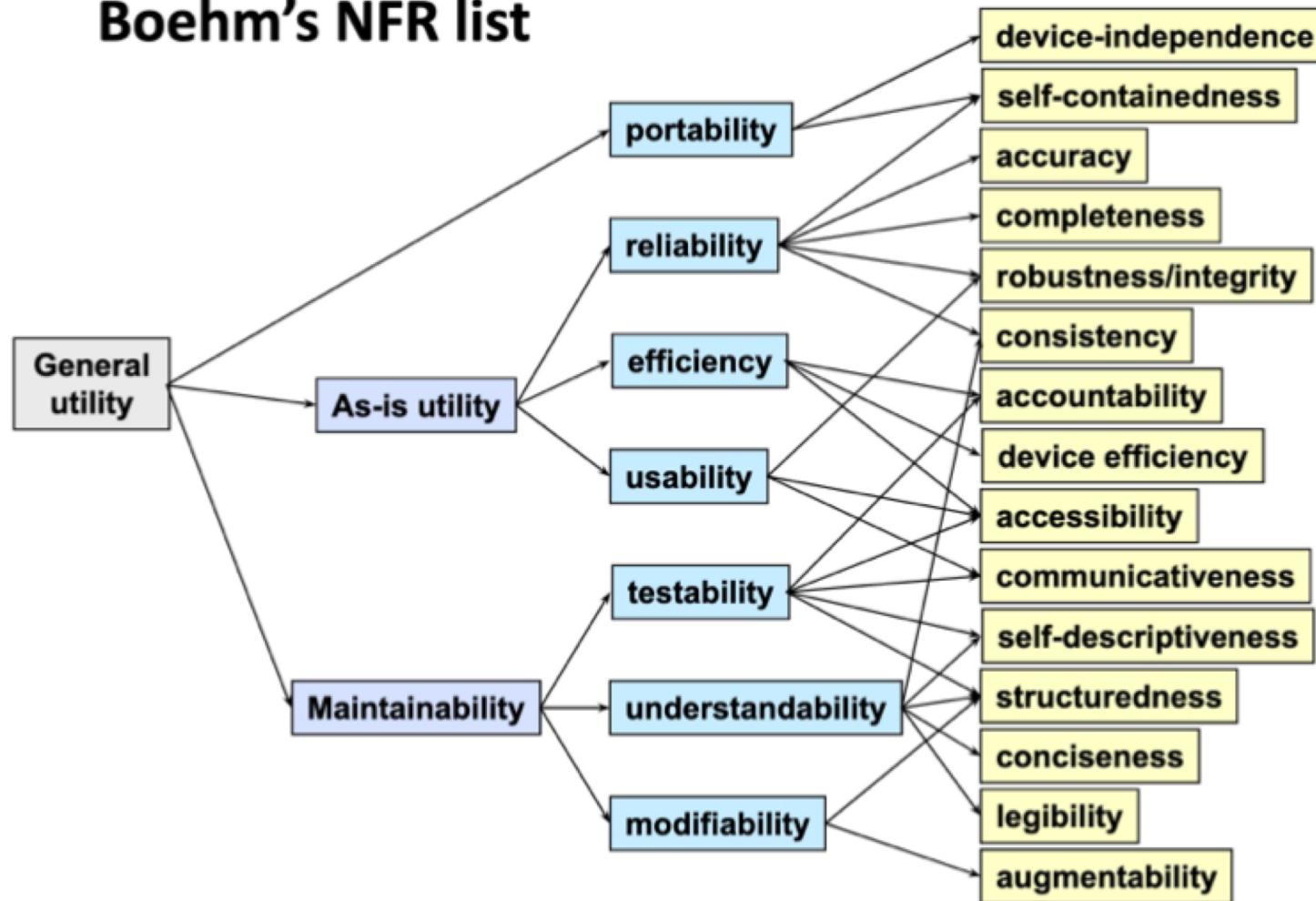
Popular Quality Attributes (1)

- » reliability
 - availability, fault tolerance, recoverability, ...
- » performance
 - time, resource utilization
- » operability
 - appropriateness recognizability, ease of use, use of interface aesthetics, technical accessibility, ...

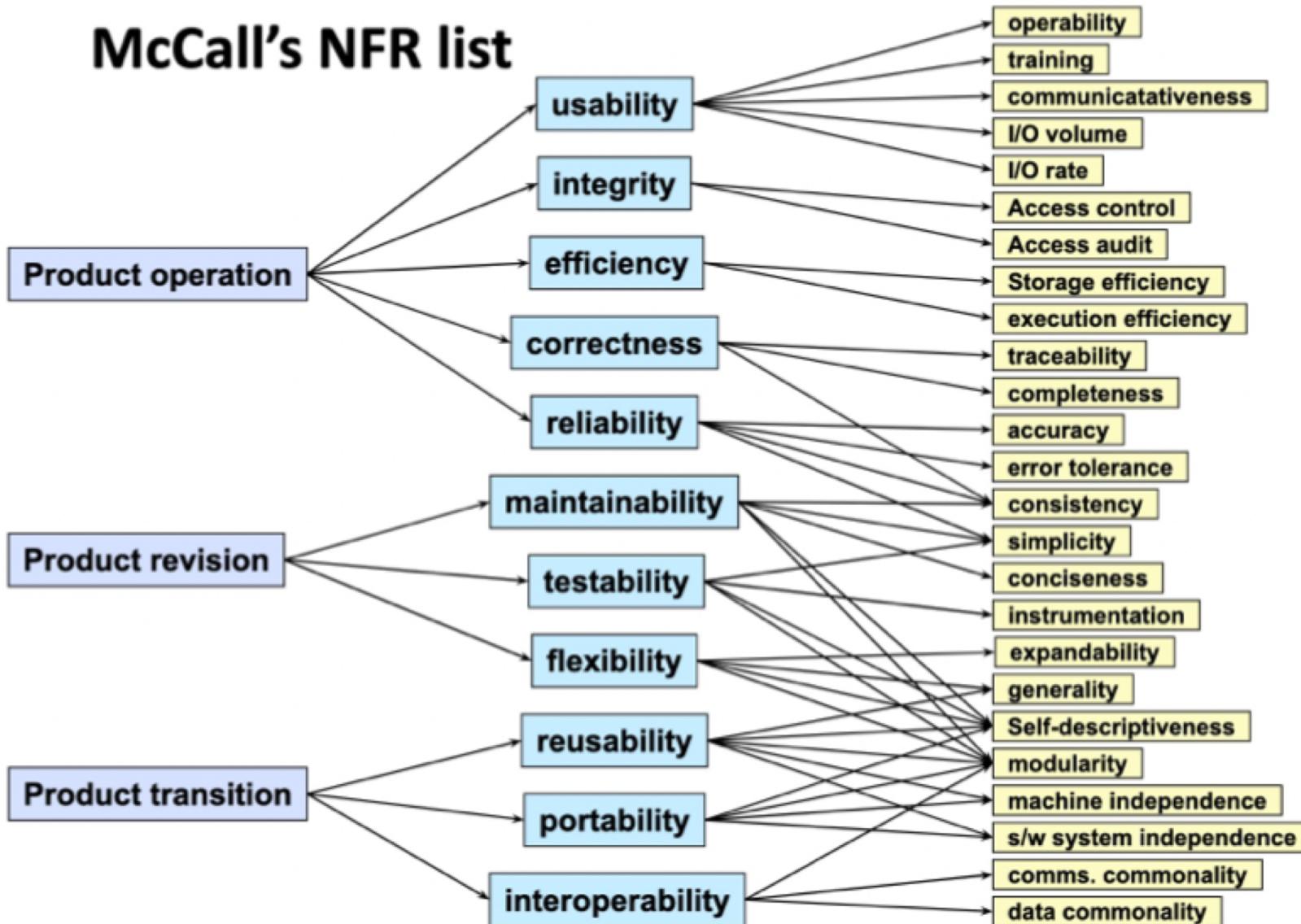
Popular Quality Attributes (2)

- » security
 - confidentiality, integrity, authenticity, ...
- » compatibility
 - co-existence, interoperability
- » maintainability
 - modularity, reusability, modifiability, testability, analyzability
- » portability
 - adaptability, replaceability, installability

Boehm's NFR list



McCall's NFR list



Il modello FURPS+ (Grady 1992): Requisiti non Funzionali

Chiamati anche requisiti di qualità, sono:

♦ **Usabilità**

facilità per l'utente di imparare ad usare il sistema, e capire il suo funzionamento. Includono:

- ♦ convenzioni adottate per le interfacce utenti
- ♦ portata dell'Help in linea
- ♦ livello della documentazione utente.

♦ **Affidabilità**

capacità di un sistema o di una componente di fornire la funzione richiesta sotto certe condizioni e per un periodo di tempo. Includono:

- ♦ un accettabile tempo medio di fallimento
- ♦ l'abilità di scoprire specificati difetti
- ♦ o di sostenere specificati attacchi alla sicurezza.

Il modello FURPS+ (Grady 1992): Requisiti non Funzionali

♦ Performance

riguardano attributi quantificabili del sistema come

- ◆ **tempo di risposta**, quanto velocemente il sistema reagisce a input utenti
- ◆ **throughput**, quanto lavoro il sistema riesce a realizzare entro un tempo specificato
- ◆ **disponibilità**, il grado di accessibilità di una componente o del sistema quando è richiesta
- ◆ **accuratezza**

♦ Supportabilità

Riguardano la semplicità di fare modifiche dopo il deployment. Includono

- ◆ **adattabilità**, l'abilità di cambiare il sistema per trattare concetti addizionali del dominio di applicazione
- ◆ **mantenibilità**, l'abilità di cambiare il sistema per trattare nuove tecnologie e per far fronte a difetti

Il modello FURPS+ (Grady 1992): Pseudo requisiti

Chiamati anche vincoli, sono:

- ◆ **Implementazione**
 - ◆ vincoli sull'implementazione del sistema, incluso l'uso di tool specifici, linguaggi di programmazione, o piattaforme hardware
- ◆ **Interfacce**
 - ◆ vincoli imposti da sistemi esterni, incluso sistemi legacy e formati di scambio
- ◆ **Operazioni**
 - ◆ vincoli sull'amministrazione e sulla gestione del sistema
- ◆ **Packaging**
 - ◆ vincoli sulla consegna reale del sistema
- ◆ **Legali**
 - ◆ riguardano licenze, regolamentazioni, e certificazioni

Identifying nonfunctional requirements (1)

Usability

- What is the level of expertise of the user?
- What user interface standards are familiar to the user?
- What documentation should be provided to the user?

Reliability

*(including robustness,
safety, and security)*

- How reliable, available, and robust should the system be?
- Is restarting the system acceptable in the event of a failure?
- How much data can the system loose?
- How should the system handle exceptions?
- Are there safety requirements of the system?
- Are there security requirements of the system ?

Performance

- How responsive should the system be?
- Are any user tasks time critical?
- How many concurrent users should it support?
- How large is a typical data store for comparable systems?
- What is the worse latency that is acceptable to users?

Supportability

*(including maintainability
and portability)*

- What are the foreseen extensions to the system?
- Who maintains the system?
- Are there plans to port the system to different software or hardware environments?

Identifying nonfunctional requirements (2)

Implementation

- Are there constraints on the hardware platform?
- Are constraints imposed by the maintenance team?
- Are constraints imposed by the testing team?

Interface

- Should the system interact with any existing systems?
- How are data exported/imported into the system?
- What standards in use by the client should be supported by the system?

Operation

- Who manages the running system?

Packaging

- Who installs the system?
- How many installations are foreseen?
- Are there time constraints on the installation?

Legal

- How should the system be licensed?
- Are any liability issues associated with system failures?
- Are any royalties or licensing fees incurred by using specific algorithms or components?

Altro esempio: Requisiti Funzionali

- ◆ *Descrivono le interazioni tra il sistema e il suo ambiente indipendentemente dalla sua implementazione (l'ambiente include l'utente e ogni altro sistema esterno)*

SatWatch è un orologio da polso che visualizza l'orario nella località corrente. SatWatch usa un sistema GPS per determinare la sua posizione e converte la posizione in un orario.

Le informazioni memorizzate in SatWath e l'accuratezza della determinazione dell'orario non richiedono che il possessore effettui un reset dell'orario. SatWatch modifica l'orario e le informazioni visualizzate quando il possessore attraversa un'area di riferimento e un confine politico.

SatWatch ha un display basato su due linee, in alto mostra l'orario (ora, minuti, secondi, zona), in basso la data (giorno, mese, anno).

Altro esempio: Requisiti non funzionali

♦ *Descrivono gli aspetti del sistema che non sono direttamente legati al comportamento (funzionalità) del sistema.*

♦ (SatWatch).

Il tempo di risposta deve essere meno di 1 secondo

L'accuratezza deve essere entro un secondo

SatWatch deve essere disponibile 24 ore al giorno eccetto dalle 2:00am-2:01am e 3:00am-3:01am

Altro esempio: Requisiti di qualità

- ◆ Ogni utente che è in grado di leggere un orologio digitale e capire le abbreviazioni relative alle zone geografiche dovrebbe essere capace di usare SatWatch (**Usabilità**)
- ◆ Poiché SatWatch non ha pulsanti, nessun fallimento che richiede di risettare l'orologio deve verificarsi (**Affidabilità**)
- ◆ SatWatch dovrebbe visualizzare la zona geografica corretta entro 5 minuti dalla fine di un periodo di blackout del sistema GPS (**Performance**)
- ◆ SatWatch dovrebbe accettare aggiornamenti attraverso l'interfaccia seriale Webify Watch (**Supportabilità**)

Altro esempio: Vincoli

- ◆ Tutto il software associato a SatWatch, incluso il software di bordo, dovrà essere scritto usando Java, per conformarsi alla politica corrente della compagnia (**Implementazione**)
- ◆ SatWatch si conforma alle interfacce fisica, elettrica, e software definite da WebifyWatch API 2.0. (**Interfacce**)

Nonfunctional Requirements

(Questions to overcome ‘Writers block’)

User interface and human factors

- ♦ **What type of user will be using the system?**
- ♦ **Will more than one type of user be using the system?**
- ♦ **What training will be required for each type of user?**
- ♦ **Is it important that the system is easy to learn?**
- ♦ **Should users be protected from making errors?**
- ♦ **What input/output devices are available**

Documentation

- ♦ **What kind of documentation is required?**
- ♦ **What audience is to be addressed by each document?**

Nonfunctional Requirements

Hardware considerations

- ♦ **What hardware is the proposed system to be used on?**
- ♦ **What are the characteristics of the target hardware, including memory size and auxiliary storage space?**

Performance characteristics

- ♦ **Are there speed, throughput, response time constraints on the system?**
- ♦ **Are there size or capacity constraints on the data to be processed by the system?**

Error handling and extreme conditions

- ♦ **How should the system respond to input errors?**
- ♦ **How should the system respond to extreme conditions?**

Nonfunctional Requirements

System interfacing

- ♦ Is input coming from systems outside the proposed system?
- ♦ Is output going to systems outside the proposed system?
- ♦ Are there restrictions on the format or medium that must be used for input or output?

Quality issues

- ♦ What are the requirements for reliability?
- ♦ Must the system trap faults?
- ♦ What is the time for restarting the system after a failure?
- ♦ Is there an acceptable downtime per 24-hour period?
- ♦ Is it important that the system be portable?

Nonfunctional Requirements

System Modifications

- ◆ **What parts of the system are likely to be modified?**
- ◆ **What sorts of modifications are expected?**

Physical Environment

- ◆ **Where will the target equipment operate?**
- ◆ **Is the target equipment in one or several locations?**
- ◆ **Will the environmental conditions be ordinary?**

Security Issues

- ◆ **Must access to data or the system be controlled?**
- ◆ **Is physical security an issue?**

Nonfunctional Requirements

Resources and Management Issues

- ◆ How often will the system be backed up?
- ◆ Who will be responsible for the back up?
- ◆ Who is responsible for system installation?
- ◆ Who will be responsible for system maintenance?

Documento di Analisi dei Requisiti (RAD)

- ◆ I risultati della raccolta dei requisiti e dell'analisi sono documentati nel RAD
- ◆ Il RAD descrive completamente il sistema in termini di requisiti funzionali e non funzionali e serve come base del contratto tra cliente e sviluppatori.
- ◆ I partecipanti coinvolti sono: cliente, utenti, project manager, analisti del sistema, progettisti
- ◆ La prima parte del documento, che include Use Case e requisiti non funzionali, è scritto durante la raccolta dei requisiti
- ◆ La formalizzazione della specifica in termini di modelli degli oggetti è scritto durante l'analisi

Documento di Analisi dei Requisiti (RAD)

1. Introduction

- 1.1. Purpose of the system
- 1.2. Scope of the system
- 1.3. Objectives and success criteria of the project
- 1.4. Definition, acronyms, and abbreviations
- 1.5. References
- 1.6. Overview

2. Current system

3. Proposed system

- 3.1. Overview
- 3.2. Functional requirements
- 3.3. Nonfunctional requirements
- 3.4. System models
 - 3.4.1. Scenarios
 - 3.4.2. Use case model
 - 3.4.3. *Object model (during analysis)*
 - 3.4.4. *Dynamic model (during analysis)*
 - 3.4.5. User interface – navigational path and screen mock-up

4. Glossary

Requirements Elicitation (Summary): Activities

- ◆ 1. Identify actors
 - Identify the different types of users of the future system
- ◆ 2. Identify scenarios
 - Identify scenarios for typical functionalities
- ◆ 3. Identify use cases
 - Abstract scenarios into use cases
- ◆ 4. Identify nonfunctional requirements
 - Identify aspects visible to the user but not directly related to functionalities
- ◆ 5. Refine use cases
 - Is the system specification complete (e.g., exceptional conditions)
- ◆ 6. Identify relationships among use cases
 - Consolidate the use case model by eliminating redundancies

A summary

- ◆ The requirements process consists of requirements elicitation and analysis.
- ◆ The requirements elicitation activity is different for:
 - ◆ **Greenfield Engineering, Reengineering, Interface Engineering**
- ◆ Scenarios:
 - ◆ **Great way to establish communication with client**
 - ◆ **Different types of scenarios: As-Is, visionary, evaluation and training**
 - ◆ **Usecases: Abstraction of scenarios**
- ◆ Pure functional decomposition is bad:
 - ◆ **Leads to unmaintainable code**
- ◆ Pure object identification is bad:
 - ◆ **May lead to wrong objects, wrong attributes, wrong methods**
- ◆ The key to successful analysis:
 - ◆ **Start with use cases and then find the participating objects**
 - ◆ **If somebody asks “What is this?”, do not answer right away. Return the question or observe the end user: “What is it used for?”**